



Commands + Code + Text Run all

RAM Disk



```
[56] from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[57] import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
[58] df=pd.read_csv('/content/drive/MyDrive/project/data - data (1).csv')
```

```
[59] df.head()
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	state
0	5/2/2014 0:00	313000.0	3	1.50	1340.0	7912.0	1.5	0	0	3	1340.0	0	1955.0	2005	18810 Densmore Ave N	Shoreline	WA
1	5/2/2014 0:00	2384000.0	5	2.50	3650.0	9050.0	2.0	0	4	5	3370.0	280	1921.0	0	709 W Blaine St	Seattle	WA
2	5/2/2014 0:00	342000.0	3	2.00	1930.0	11947.0	1.0	0	0	4	1930.0	0	1966.0	0	26206-26214 143rd Ave SE	Kent	WA
3	5/2/2014 0:00	420000.0	3	2.25	2000.0	8030.0	1.0	0	0	4	1000.0	1000	1963.0	0	857 170th Pl NE	Bellevue	WA
4	5/2/2014 0:00	550000.0	4	2.50	NaN	10500.0	1.0	0	0	4	1140.0	800	1976.0	1992	9105 170th Ave NE	Redmond	WA

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
[60] df.dtypes
```

	0
date	object
price	float64
bedrooms	int64
bathrooms	float64
sqft_living	float64
sqft_lot	float64
floors	float64
waterfront	int64
view	int64
condition	int64
sqft_above	float64
sqft_basement	int64
yr_built	float64
yr_renovated	int64
street	object
city	object
statezip	object
country	object

dtype: object

```
[61] df.describe()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated
count	4.600000e+03	4600.000000	4580.000000	4579.000000	4.586000e+03	4600.000000	4600.000000	4600.000000	4600.000000	4585.000000	4600.000000	4582.000000	4600.000000
mean	5.519630e+05	3.400870	2.161517	2138.905001	1.487521e+04	1.512065	0.007174	0.240652	3.451739	1828.303381	312.081522	1970.849192	808.608261
std	5.638347e+05	0.908848	0.784793	963.148687	3.593585e+04	0.538288	0.084404	0.778405	0.677230	862.814307	464.137228	29.716777	979.414536
min	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.000000	0.000000	1.000000	370.000000	0.000000	1900.000000	0.000000
25%	3.228750e+05	3.000000	1.750000	1460.000000	5.001000e+03	1.000000	0.000000	0.000000	3.000000	1190.000000	0.000000	1951.000000	0.000000
50%	4.609435e+05	3.000000	2.250000	1980.000000	7.684500e+03	1.500000	0.000000	0.000000	3.000000	1590.000000	0.000000	1976.000000	0.000000
75%	6.549625e+05	4.000000	2.500000	2620.000000	1.100375e+04	2.000000	0.000000	0.000000	4.000000	2300.000000	610.000000	1997.000000	1999.000000
max	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.000000	4.000000	5.000000	9410.000000	4820.000000	2014.000000	2014.000000

```
[62] df.columns
```

Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'street', 'city', 'statezip', 'country'], dtype='object')

	0
date	0
price	0
bedrooms	0
bathrooms	20
sqft_living	21
sqft_lot	14
floors	0
waterfront	0
view	0
condition	0
sqft_above	15
sqft_basement	0
yr_built	18
yr_renovated	0
street	0
city	0
statezip	0
country	0

dtype: int64

```
[64] missing_percentage = df.isnull().mean() * 1100
      threshold = 30
```

```
[65] columns_to_drop = missing_percentage[missing_percentage>threshold].index
```

```
[66] df=df.drop(columns=columns_to_drop)
      print(f'columns dropped :{list (columns_to_drop)}')
      print(f'remaining columns : {df.columns}')
```

```
columns dropped :[]
remaining columns : Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
                          'floors', 'waterfront', 'view', 'condition', 'sqft_above',
                          'sqft_basement', 'yr_built', 'yr_renovated', 'street', 'city',
                          'statezip', 'country'],
                          dtype='object')
```

```
[67] from sklearn.preprocessing import LabelEncoder
      label_encoder = LabelEncoder()
      for column in df.select_dtypes(include=['object']).columns:
          df[column] = label_encoder.fit_transform(df[column])
      df.head()
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	statezip	country
0	10	313000.0	3	1.50	1340.0	7912.0	1.5	0	0	3	1340.0	0	1955.0	2005	1522	36	62	
1	10	2384000.0	5	2.50	3650.0	9050.0	2.0	0	4	5	3370.0	280	1921.0	0	3899	35	58	
2	10	342000.0	3	2.00	1930.0	11947.0	1.0	0	0	4	1930.0	0	1966.0	0	2291	18	26	
3	10	420000.0	3	2.25	2000.0	8030.0	1.0	0	0	4	1000.0	1000	1963.0	0	4263	3	7	
4	10	550000.0	4	2.50	NaN	10500.0	1.0	0	0	4	1140.0	800	1976.0	1992	4352	31	31	

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
[68] data=df.copy()
      def detect_outliers_iqr(column):
          Q1 = column.quantile(0.25)
          Q3 = column.quantile(0.75)
          IQR = Q3 - Q1
          lower_bound = Q1 - 1.5 * IQR
          upper_bound = Q3 + 1.5 * IQR
          return (column < lower_bound) | (column > upper_bound)
          for col in data.columns:
              data[f'{col}_outlier'] = detect_outliers_iqr(data[col])
          outliers_only = data[data.filter(like='_outlier').any(axis=1)]
          outliers_only
```

```
[69] for column in df.select_dtypes(include=['float64']).columns:
      mean_value = df[column].mean()
      df[column] = df[column].fillna(mean_value)
      for column in df.select_dtypes(include=['object']).columns:
          mode_value = df[column].mode()[0]
          df[column] = df[column].fillna(mode_value)
      df.isnull().sum()
```

	0
date	0
price	0
bedrooms	0
bathrooms	0
sqft_living	0
sqft_lot	0
floors	0
waterfront	0
view	0

```

sqft_above 0
sqft_basement 0
yr_built 0
yr_renovated 0
street 0
city 0
statezip 0
country 0

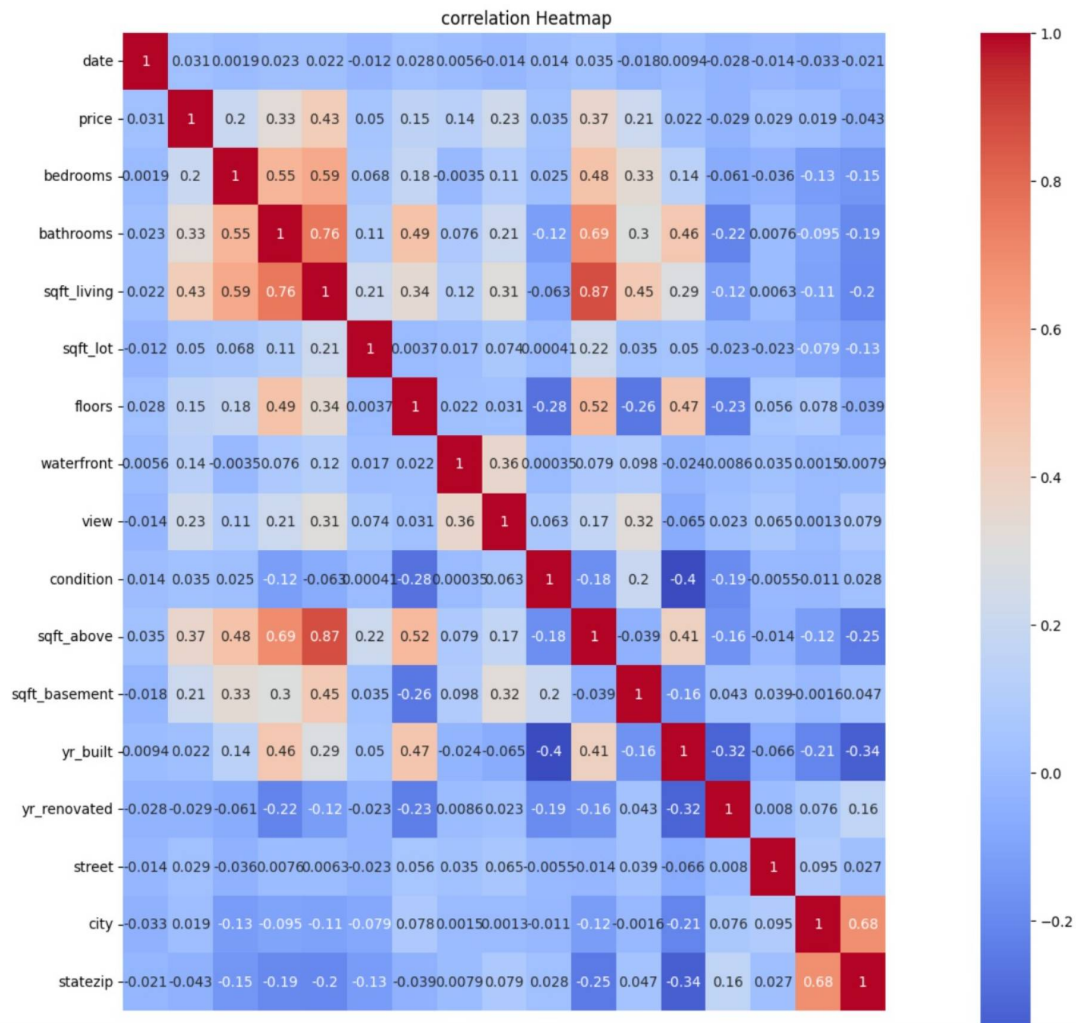
```

dtype: int64

```

[76]
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('correlation Heatmap')
plt.show()

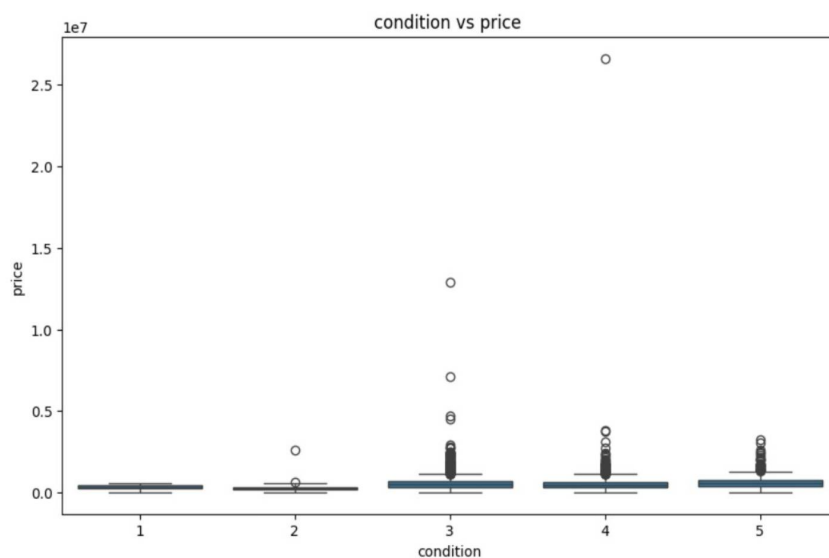
```



```

[71]
sns.boxplot(data=df, x='condition', y='price')
plt.title('condition vs price')
plt.show()

```

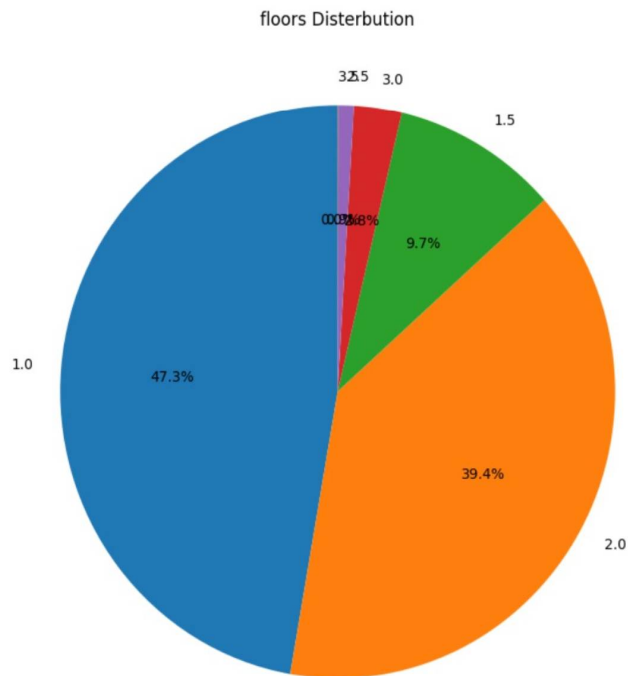


```

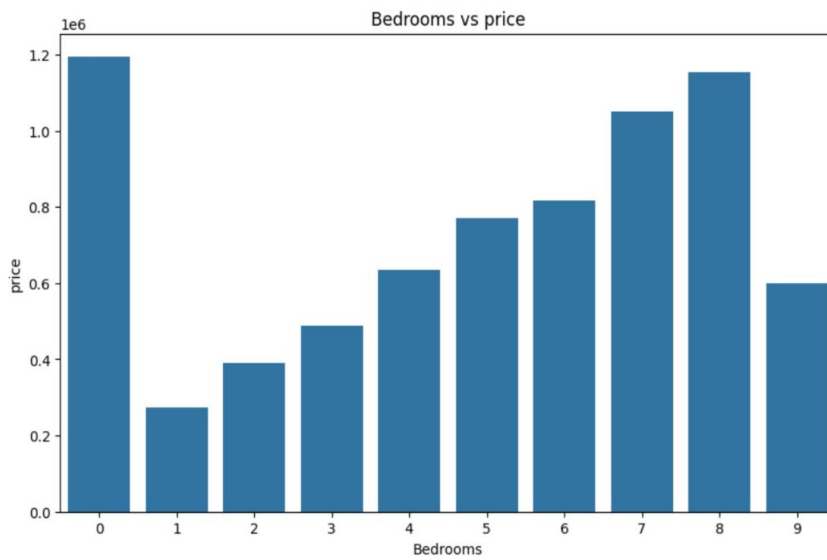
[77]
floor_counts = df['floors'].value_counts()
plt.figure(figsize=(9,9))
plt.pie(floor_counts, labels=floor counts.index,

```

```
plt.title('floors Disterbution')
plt.show()
```



```
[73]
✓ 0s
plt.figure(figsize=(10,6))
sns.barplot(data=df , x='bedrooms',
y='price' ,estimator= "mean"
,ci=None)
plt.title('Bedrooms vs price')
plt.xlabel('Bedrooms')
plt.ylabel('price')
plt.show()
```



<https://www.kaggle.com/datasets/shree1992/housedata>

